Часть III

Структурный подход в распознавании образов (I)

Разделы

Разделы

Распознавание образов: дискриминантный подход

① Дискриминантный (признаковый) подход

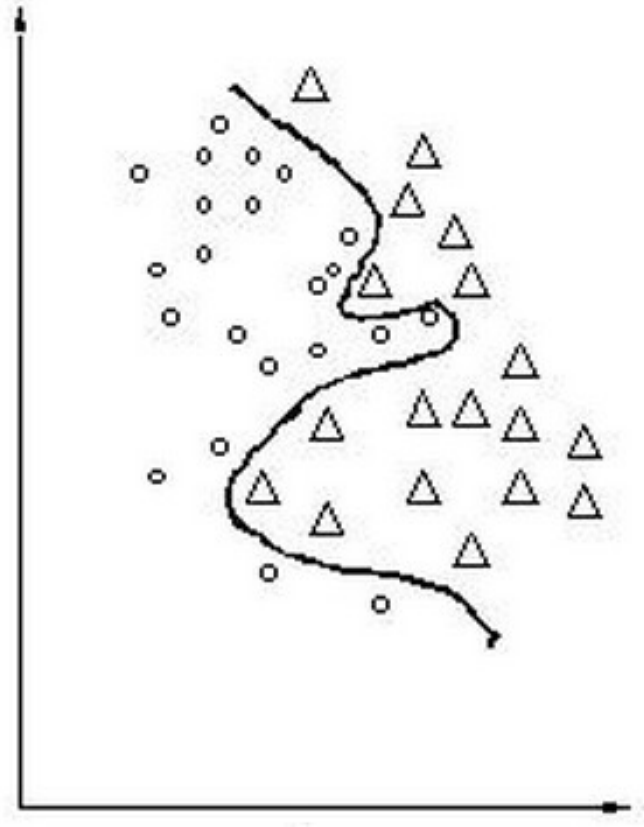
Объекты характеризуются набором признаков и распознавание проводят разбиением пространства признаков на области.

Методы:

- метрические (NN, ...);
- разделяющие поверхности (SVM, ...);
- потенциальные функции;
- логические;
- информационные;
- коллективные решающие правила (*области компетенции, голосование, алгебраический подход*);
- ...

Математический аппарат — метрические теории, логические функции, теория информации, алгебра алгоритмов, ...

Распознавание образов: дискриминантный подход...



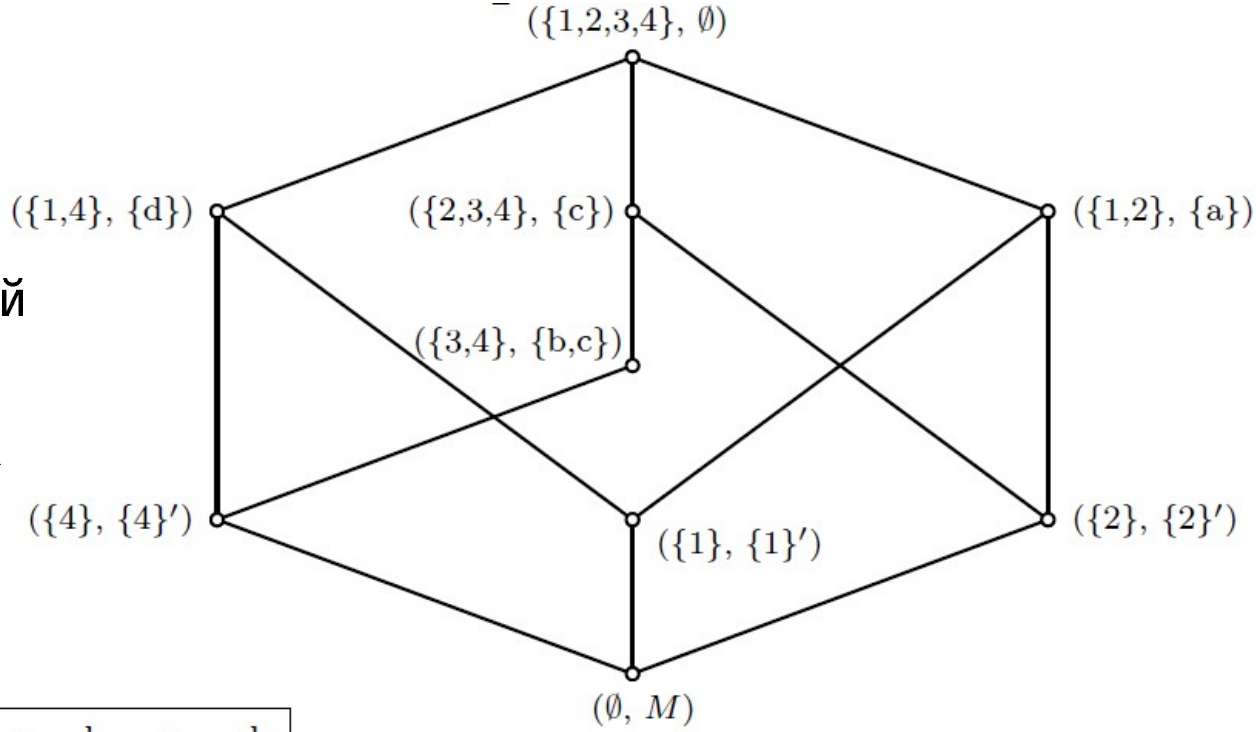
«Ленивые» и «не-ленивые» алгоритмы распознавания

При этом игнорируется информация:
(1) структурная; (2) объект-признак

Распознавание образов: реляционный подход

② Реляционный подход — анализ формальных понятий (АФП): формализация понятия в виде пары объём-содержание.

Математический аппарат — теория решёток



	G \ M	a	b	c	d
1	△	×			×
2	◺	×		×	
3	▭		×	×	
4	□		×	×	×

a – ровно 3 вершины,
b – ровно 4 вершины,
c – имеет прямой угол,
d – все стороны равны

Распознавание образов: синтаксический подход

③ Синтаксический (структурный, лингвистический) подход.

Структурный подход применяется к задачам, в которых:

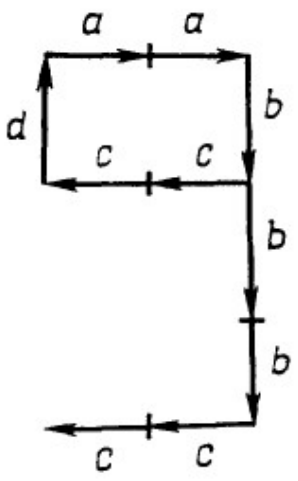
1) важна информация об объекте, как состоящего из элементарных частей, связанных между собой определённым образом;
2) требуется, чтобы она давала возможность не только классифицировать объект, но и причины, которые исключают его отнесение к другому классу.

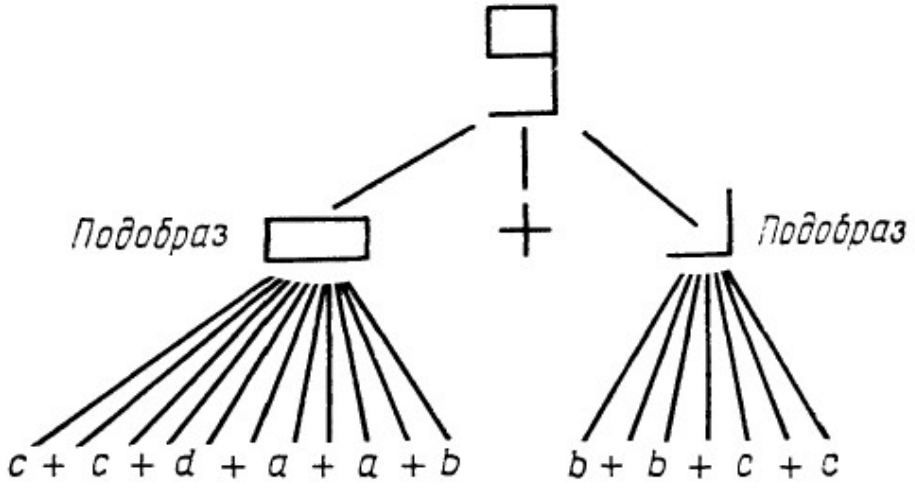
Примеры — распознавание

- изображений и анализ сцен: объекты обычно сложны, число требуемых признаков часто велико $\Rightarrow$ удобство описания их через подобразы;
- сигналов: акустических (речь), электрических (ЭКГ, ЭЭГ, ...), электромагнитных (отражённые радиолокационные), оптических (изображения), ...

Распознавание образов: синтаксический подход...

Объект представляется состоящим из неделимых частей — примитивов, объединяющихся по определённым правилам.





Разные правила порождают разные классы объектов.

Лингвистическая интерпретация: аналогия между структурой объекта и синтаксисом языка даёт возможность использовать аппарат *математической лингвистики*.

Распознавание образов: синтаксический подход...

При дискриминантном подходе надо

1) сформировать признаковое пространство;
2) для каждого класса иметь достаточное число прецедентов.

При синтаксическом подходе надо задать правило построения классов.

Понятно, что синтаксический подход эффективен, когда:

1) имеется небольшое количество примитивов (элементарных подобъектов) которые легко вычленяются;
2) построение образов из примитивов описывается сравнительно простыми правилами.

Синтаксическое распознавание: составляющие подхода

На примере распознавания/обработки изображений.

① Предобработка:

- кодирование и аппроксимация — представление в виде, удобном для дальнейшей обработки.

  Примеры:
  - бинаризация черно-белых изображений,
  - дискредитация непрерывного сигнала,
  - представление функции конечным набором коэффициентов разложения по некоторой системе функций (Фурье, Уолша, ...).

- фильтрации, восстановление и улучшения объекта — ликвидация шума, восстановления искажений, улучшения качества.

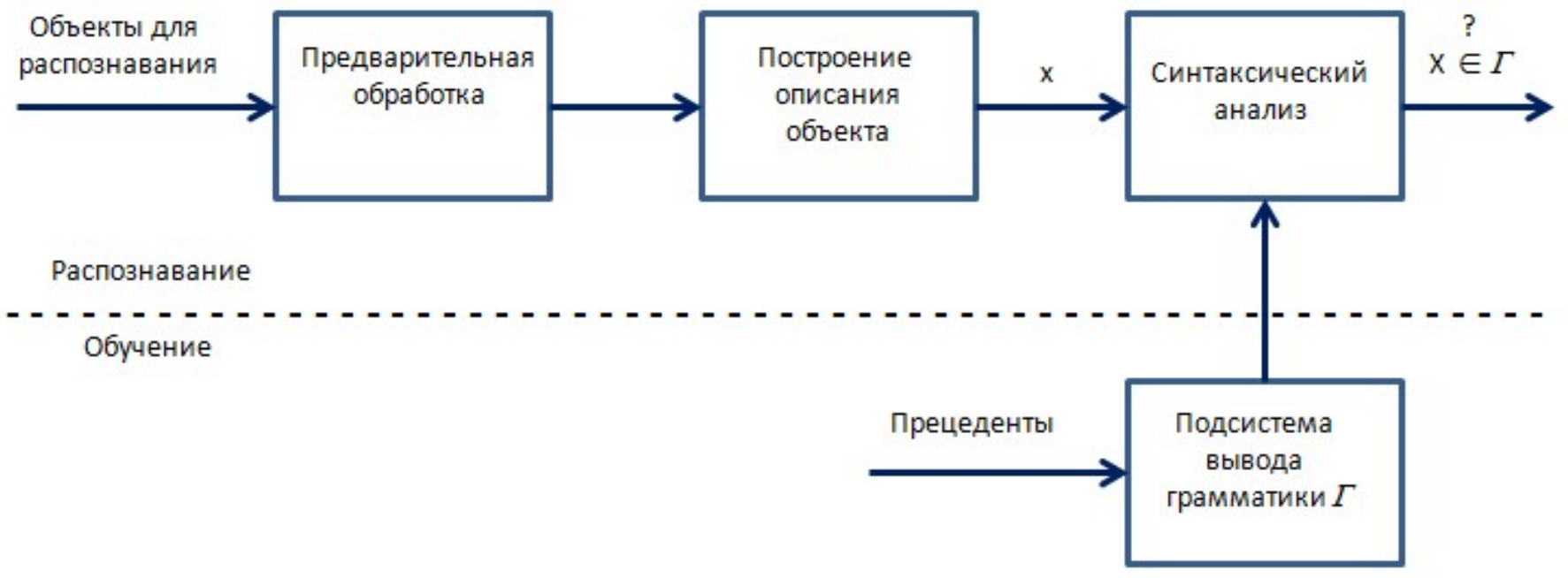
Синтаксическое распознавание: составляющие подхода...

② Построение описания объекта на основе заранее заданных синтаксических операций:

- выделение примитивов — элементарных частей объектов;
- представление объекта в виде композиции примитивов в соответствии с той или иной структурой.

③ Синтаксический анализ (грамматический разбор) — получение синтаксического описания объекта через примитивы — построение дерева грамматического разбора;

Синтаксическое распознавание: схема подхода



Структурный подход обеспечивает высокое быстродействие: распознавание сводится к сравнению символьных описаний структур, а не исходных изображений $\Rightarrow$ преимущество при поиске в больших коллекциях графических документов.

Разделы

Методы предобработки: кодирование и аппроксимация

1. Дискретизация — взятие измерений (отсчетов) в дискретные моменты времени.

Теорема (Котельникова-Найквиста-Шеннона)
Если преобразование Фурье функции $f(t)$

$$\mathcal{F}\{f(t)\} = \frac{1}{\sqrt{2\pi}} \int\limits_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

равно нулю вне интервала $-\omega_{max} \leqslant \omega \leqslant +\omega_{max}$, *то функция* $f(t)$ *может быть точно восстановлена по её значениям, взятым на расстоянии интервала дискретизации* $\triangle t \leqslant \triangle t_{max} = 1/(2f_{max})$, $\omega = 2\pi f$ *друг от друга в виде* sinc *-ряда Котельникова:*

$$f(t) = \sum_{n=-\infty}^{+\infty} f(n \triangle t) \cdot \mathrm{sinc}\left(\frac{t}{\triangle t} - n\right), \qquad \mathrm{sinc}\ x = \frac{\sin \pi x}{\pi x}.$$

Теорема Котельникова: замечания

- Теорема Котельникова применима лишь для сигналов с ограниченным спектром, т.е. принципиально для сигналов бесконечных во времени.
- На практике: полагая $\widehat{\mathcal{F}} = 0$, когда $\mathcal{F}\{f(t)\} \approx 0 \Rightarrow$, при восстановлении функции по отсчётам будет получена лишь аппроксимация $f(t)$.
- Локальная интерполяция (по ограниченному числу точек) и недостаточная частота дискретизации вносит высокочастотные искажения в виде периодических копий полезной полосы частот (алиасинг, aliasing).

Теорема Котельникова: замечания...

- Сигнал невозможно восстанавливать по мере его получения и, следовательно
  - высококачественная дискредитация sinc-рядом возможна только при offline-обработке;
  - при online-обработке приходится запоминать некоторое количество отсчетов и производить интерполяцию с незначительной задержкой.
- Имеется аналогичная теорема для двумерного случая — функции $f(x, y)$.

Теорема Котельникова: рекомендации

На практике ЦОС необходимо придерживаться следующих основных рекомендаций:

- частота дискретизации должна выбираться так, чтобы она не менее чем в 2-3 раза превышала верхнюю частоту информативных составляющих сигнала (oversampling);
- предварительно сигнал следует отфильтровать фильтром нижних частот с частотой среза намного меньшей, чем половина выбранной частоты дискретизации.

Методы предобработки...

2. Квантование — присваивание цифровому отсчету значения, соответствующего некоторому фиксированному уровню сигнала. *Уровень квантования* — уровень, превышение которого по абсолютной величине приводит к переходу на следующий *шаг квантования*: $f(t) \approx \widehat{f}(t) \in \{$*уровни квантования*$\}$, которые могут располагаться

- равномерно — простота представления; при шаге квантования a значения ошибки квантования:
  максимальная — $\pm a/2$; с.к.о. — $\frac{a}{\sqrt{12}} \approx 0{,}29a$.
- неравномерно — выигрыш в точности представления: если, например, уровни яркости изображения в определенном диапазоне встречаются чаще, чем в других диапазонах, то при более плотном расположении в нём уровней квантования, точность представления увеличится.

Методы предобработки: кодирование квантованных сигналов I

После дискретизации и квантования объекта уровень квантования в каждой точке отсчета кодируют.

Характеристики кода:

- основание кода q — число цифр, букв, из которых строится код (обычно $q = 2$);
- m — число возможных уровней квантования; для представления любого заданного уровня необходимо по крайне мере $\log_2 m$ бит.
  Если вероятность появления каждого из возможных уровней известна заранее, то можно построить более эффективный код, снизив среднее число бит на один уровень. Основная идея: использовать короткие коды для наиболее часто встречающихся уровней квантования.

Методы предобработки: кодирование квантованных сигналов II

- длина кода n, число информационных разрядов k, число проверочных разрядов $m = n - k$;
- избыточность кода $R = m/n$ (для разделимых блочных кодов);
- кодовое расстояние d — минимальное число разрядов с различными символами;
- характеристики помехоустойчивости избыточных кодов, вероятности необнаружения ошибки и т.д.

Методы предобработки: фильтрация

3. Фильтрация, восстановление и улучшение — изменение частотного состава (спектра) сигнала.
Результат — повышение качества данных $\Rightarrow$ облегчение обнаружения заданных элементов (сигнала, изображения).

Для фильтрации используют операции, инвариантные ко времени и к изменению положения: «сглаживание», «повышение контраста», ...

Свёртка с окном $g(x, y)$, $(u, v) \notin D \Rightarrow g(x, y) = 0$:

$$f(x, y) \mapsto f_g(x, y) = (f * g)(x, y) = \int\limits_{(u,v)\in D} g(x-u,\, y-v) f(u, v)\, du\, dv$$

Методы предобработки: сглаживание...

1. Сглаживание — подавление шума, усреднение значений по некоторой окрестности

$$g(x,y) = \frac{1}{|D|}, \quad g(x,y) = \frac{1}{div} \cdot \begin{array}{|c|c|c|}\hline 1/2 & 3/4 & 1/2 \\ \hline 3/4 & 1 & 3/4 \\ \hline 1/2 & 3/4 & 1/2 \\ \hline \end{array}, \; div = 6$$

div — коэффициент нормирования, для того чтобы средняя интенсивность оставалась не изменой.

2. Повышение контраста (дифференцирование)

$\nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ — свёртка с окном

$$g(x,y) = \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline -2 & +12 & -2 \\ \hline -1 & -2 & -1 \\ \hline \end{array}, \qquad g(x,y) = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & +5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

Методы предобработки: фильтрация...

Для дискретных сигналов применяют *линейную цифровую фильтрацию*, задаваемую *обобщенным разностным уравнением цифрового фильтра*:

$$y(n) = \sum_{i \in I} b_i x(n-i) + \sum_{j \in J} a_j y(n-j),$$

где $x(n)$ и $y(n)$ — соответственно входной и выходной сигнал,
b_i, a_j — *коэффициенты фильтра*,
I, J — интервалы $\mathbb{Z}$, содержащие 0, *параметры фильтра*.

Частные случаи:

- $a_j = 0$ для всех $j \in J$ — *нерекурсивный* фильтр,
  иначе — *рекурсивный*;
- все коэффициенты фильтра суть константы — *фильтр с постоянными коэффициентами*, иначе — *адаптивный*.

Цифровые фильтры: примеры

1) Сглаживание пятёрками —

$$y(n) = \frac{1}{5}\left[x(n-2) + x(n-1) + x(n) + x(n+1) + x(n+2)\right]$$

2) Фильтр нижних частот (*фильтр Хемминга*) —

$$y(n) = \frac{1}{8}x(n) + \frac{1}{4}x(n-1) + \frac{1}{4}x(n-2) + \frac{1}{4}x(n-3) + \frac{1}{8}x(n-4)$$

3) Сглаживающий дифференциатор —

$$y(n) = \frac{1}{8}\left[x(n-8) + x(n-7) + x(n-6) + x(n-5) - \right.$$
$$\left. - x(n-3) - x(n-2) - x(n-1) - x(n)\right]$$

4) Численное интегрирование по методу трапеций —

$$y(n) = y(n-1) + \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$$

Предобработка: сегментация изображений

4. Сегментация — разбиение изображения на фрагменты, не похожие по какому-то признаку.
Сегментация разделяет изображение на составляющие части или объекты, а уровень детализации разделяемых областей зависит от решаемой задачи $\Rightarrow$ универсального метода сегментации не существует.

Задачи автоматической сегментации изображений:

1) выделение областей с известными свойствами;
2) разбиение изображения на однородные области.

Предобработка: методы сегментации изображений

— базируются на распределении яркостей точек изображения.

① Пороговые методы —
используют *пороговую операцию*: фрагмент есть совокупность точек, яркость которых больше/меньше порога/лежит между пороговыми значениями).

Например, в задаче распознавания знаков на бумаге, сами знаки представлены совокупностью черных, а бумага (фон) — совокупностью белых точек $\Rightarrow$ отделить знак от фона можно при помощи пороговой операции.

Существенная трудность — выбор значения порога.

Пороговые методы сегментации изображений

- *Метод процентúлей.*
  Если определена приблизительная площадь S фрагмента, то выбирают пороги так, чтобы площадь выделяемого сегмента с яркостью меньше/больше порога была приблизительно равна S.

- *Метод моды.*
  Если определен интервал яркости I фрагмента, то выбирают пороги так, чтобы в сегмент попали точки с яркостью из I.
  Например, если яркости фрагмента имеет пик, то границами I могут быть локальные минимумы распределения, расположенным по обе стороны от этого пика.

Пороговые методы сегментации изображений...

- *Метод Оцу (Otsu)* вычисления порога бинаризации полутонового изображения: ищется порог h, минимизирующий дисперсию внутри классов:

$$h = \arg\min_t \left\{\omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)\right\},$$

где веса ω_i — вероятности (доли) двух классов, разделенных порогом t, σ_i^2 — дисперсии классов, $i = 1, 2$.

Минимизация дисперсии внутри класса = максимизация дисперсии между классами.

Пороговые методы сегментации изображений...

- *Прослеживание границ (контуров, краёв, ...).*
  Например,
  - простой способ отслеживания границ на двухградационном изображении: просмотр изображения, пока не встретится пара точек растра разной яркости,
  - управляемые (steerable) фильтры, осуществлящие дифференцирование по направлению.

  Для повышения устойчивости к шуму, перед применением фильтрации изображение обычно размывают.

Пороговые методы сегментации изображений...

Пример оператора пространственного дифференцирования для формирования контурных изображений — *оператор Собела*, имеющий наименьший коэффициент утолщения контурной линии.

$f(x,y) \to g(x,y) = \|\nabla f(x,y)\| = \sqrt{d_1^2 + d_2^2}$ — модуль градиента

$d_1,\ d_2$ — суммы элементов свёртки $f(x,y)$ с окнами H_1 и H_2 соответственно

$$H_1 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad H_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

«Не-пороговые» методы сегментации изображений

② «Не-пороговые» методы

- *Выращивание регионов, дробление-слияние*
  — последовательная проверка точек на их принадлежность к фрагменту.
  Методы этой группы учитывают пространственное расположение точек напрямую и позволяют:
  - выделять фрагменты;
  - прослеживать границы или контуры фрагментов;
  - получать характеристики фрагментов (площадь, параметры формы, ...).

  Методы позволяет варьировать значение порога в зависимости от характера уже проверенных точеки результат такой процедуры зависит от того, какая последовательность будет выбрана.

«Не-пороговые» методы сегментации изображений...

- *Моделирование изображения случайным полем*
  — в основе предположение, что характеристики каждой точки изображения зависит от характеристик некоторого множества соседних точек.
  Как правило, рассматривают марковское случайное поле.

- *Методы, основанные на теории графов*
  Общая идея:
  - изображение представляется в виде взвешенного графа, с вершинами в точках изображения,
  - вес ребра графа отражает сходство точек в некотором смысле (расстояние между точками по некоторой метрике),
  - разбиение изображения моделируется разрезами графа.

«Не-пороговые» методы сегментации изображений...

- *Оптимизационный подход*
  — задачу разбиения изображения на однородные области сводят к задаче оптимизации некоторого функционала от разбиения.
  Например, в графовых методах оптимизируется функционал стоимости разреза.

- *Настройка параметров*
  — одно из наиболее перспективных на данный момент направлений.
  Почти у всех вышеописанных методов сегментации есть масса параметров, значения которых для каждой конкретной задачи приходится подбирать эвристически.
  Одним из подходов к решению этой проблемы является использование машинного обучения.

Разделы

Выбор терминальных элементов

— первый этап построения синтаксической модели образов.

Требования к терминальным элементам (примитивам):

- они должны служить основными элементами образов и обеспечивать адекватное описание объекта в терминах заданных структурных отношений;
- их выделение и распознавание должны легко осуществляться.

Эти требования часто противоречат друг другу.

Примеры непроизводных элементов:

- для речевых образов — совокупность фонем (= звуков, минимальных различимых единиц языка).
- для рукописного текста — штрихи.

Разные постановки задачи обуславливают выбор разных непроизводных элементов.

Языки и порождающие грамматики (напоминание)

Определение

Порождающая грамматика G *есть четверка*

$$G = \langle V, W, I, R \rangle,$$

где (все множества конечны)

V — *множество терминальных символов;*

W — *множество нетерминальных (вспомогательных) символов,* $V \cap W = \varnothing$*;*

I — *начальный символ,* $I \in W$*;*

R — *совокупность правил вывода (подстановок цепочек символов).*

Язык $L(G)$, порождаемый грамматикой G —

$$L(G) = \{ x \in V^* \mid \exists \text{ вывод } I \Rightarrow x \}$$

Порождающие грамматики по форме правил подстановки разделены на 4 типа (Н.Хомский, 1957).

Грамматики типа 0 — *неограниченные*

Самый широкий класс: не имеет каких-либо ограничений на вид правил подстановки.

Пример ($V = \{a, b, c\}$, $W = \{I, A, B\}$)

$$
\begin{aligned}
R: I &\to aAbc \\
Ab &\to bA \\
Ac &\to Bbcc \\
bB &\to Bb \\
aB &\to aaA \\
aB &\to \quad \text{— пустой символ}
\end{aligned}
$$

Грамматика порождает слова вида $x = a^n b^{n+2} c^{n+2}$, $n \geqslant 0$.
Пример: $I \Rightarrow aAbc \Rightarrow abAc \Rightarrow abBbcc \Rightarrow aBbbcc \Rightarrow bbcc$.

Для использования этот класс слишком широк.

Грамматики типа 1 — *непосредственно составляющих*

или ***контекстно-зависимые***.

Имеют правила вида $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$,
где $A \in W,\ \beta \in T^+,\ \alpha_1, \alpha_2 \in T^*,\ T = V \cup W$.

Пример ($V = \{a, b, c\},\ W = \{I, A, B\}$)

$$
\begin{aligned}
R: I &\to abc \\
I &\to aAbc \\
Ab &\to bA \\
Ac &\to aaA \\
aB &\to aa
\end{aligned}
$$

Эта грамматика непосредственно составляющих порождает слова вида $x = a^n b^n c^n,\ n \geqslant 1$.

Грамматики типа 2 — *бесконтекстные*

или ***контекстно-свободные, КС***.

Имеют правила подстановки вида $A \to \beta$, где $A \in W$, $\beta \in T^+$, т.е. заменяют вспомогательный символ A на непустую цепочку произвольных символов вне зависимости от контекста (окружения A).

Пример ($V = \{a, b\}$, $W = \{I\}$)

$$R : I \to ab$$
$$I \to aIb$$

Эта бесконтекстная грамматика порождает слова вида $x = a^n b^n$, $n \geqslant 1$.

Грамматики типа 2...

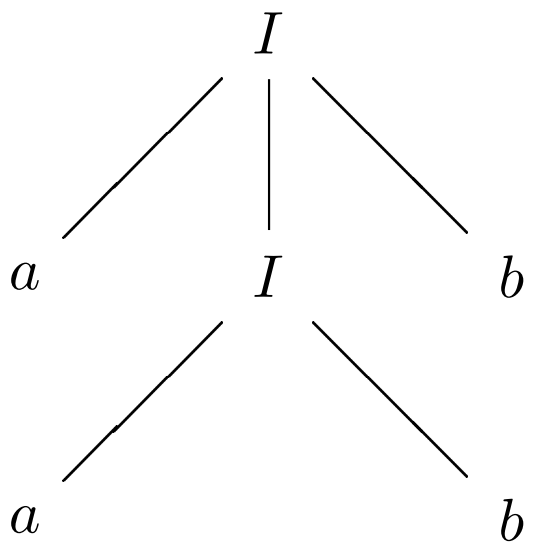
Другим методом описания вывода в бесконтекстной грамматике является использование *деревьев вывода*.

Построение дерева вывода:

- каждая вершина дерева имеет метку — символ из алфавита T, корень дерева имеет метку I.
- если вершина с меткой A — не лист дерева, то $A \in W$.
- если вершине n с меткой A прямо подчинены вершины $n_1, \ldots, n_k$, помеченные (слева направо) $A_1, \ldots, A_k$, то в R должно быть правило $A \to A_1, \ldots, A_k$.

Грамматики типа 2...

Вывод слова $aabb$:



Грамматики типа 3 — *автоматные или регулярные*

Имеют правила подстановок вида $A \to aB$ или $A \to a$, где $A, B \in W$.

Это самые простые из формальных грамматик; они являются контекстно-свободными, но с ограниченными возможностями.

Пример ($V = \{a, b\}$, $W = \{I, A\}$)

$$\begin{aligned} R: I &\to aA \\ A &\to aA \\ A &\to b \end{aligned}$$

Эта регулярная грамматика грамматика порождает слова вида $x = a^n b$, $n \geqslant 1$.

$$G_3 \subset G_2 \subset G_1 \subset G_0$$

Программные грамматики

— ограничивает выбор следующего правила подстановки.

Определение

Программная грамматика G_p *есть пятёрка*

$$G_p = \langle V, W, I, P, J \rangle,$$

где (все множества конечны)

V — *множество терминальных символов;*
W — *множество нетерминальных символов;*
I — *начальный символ,* $I \in W$*;*
P — *множество правил вывода (подстановки);*
J — *множество меток правил подстановки, обычно* $J = \{1, \ldots, m\}$ — *номер правила.*

Программные грамматики: определение

Правило вывода из P *имеет вид*

$$(r) \quad \alpha \to \beta \quad S \quad F,$$

где $r \in J$ *— метка,*

$S, F \subseteq J$ *— два списка меток переходов при успехе и неудаче соответственно,*

$\alpha \to \beta$ *— правило подстановки (ядро),*

$\alpha \in T^* W T^*,\ \beta \in T^*.$

Программная грамматика действует следующим образом:

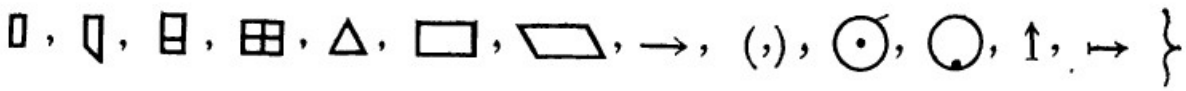
- сначала применяется правило с меткой (1);
- если делается попытка применить правило (r), то при невозможности его применения следующее выбирается из списка F, а в противном случае его выполнения следующее выбирается из списка S.

Примеры двумерных грамматик. 1. Домики

① Грамматика, описывающая изображения домиков:

$$G = \langle V, W, I, R \rangle.$$

$V = \{ \langle$домик$\rangle, \langle$профиль$\rangle, \langle$фасад$\rangle, \langle$крыша$\rangle, \langle$фронтон$\rangle,$
$\langle$стена$\rangle, \langle$дымоход$\rangle, \langle$окна$\rangle, \langle$дверь$\rangle \}, \qquad I = \langle$домик$\rangle$

$W =$ 

Обозначения:

 (XY) X — справа от Y,

 (XY) X — внутри Y,

 (XY) X — внутри Y и внизу,

 (XY) X — лежит сверху Y,

 (XY) X — прилегает к Y справа.

Примеры двумерных грамматик. 1. Домики $R =$

⟨дверь⟩ → 

⟨окна⟩ →  , ⟨окна⟩ → → (⟨окна⟩, )

⟨дымоход⟩ → , ⟨дымоход⟩ → 

⟨стена⟩ → , ⟨стена⟩ →  (⟨дверь⟩, )

⟨стена⟩ → (⟨окна⟩,)

⟨фронтон⟩ →  , ⟨фронтон⟩ →  (⟨дымоход⟩, )

⟨крыша⟩ →  , ⟨крыша⟩ →  (⟨дымоход⟩, )

⟨фасад⟩ →  (⟨фронтон⟩, ⟨стена⟩)

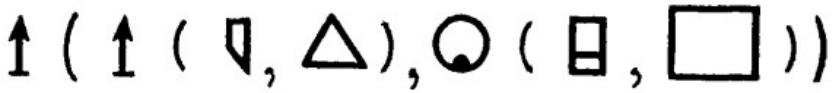
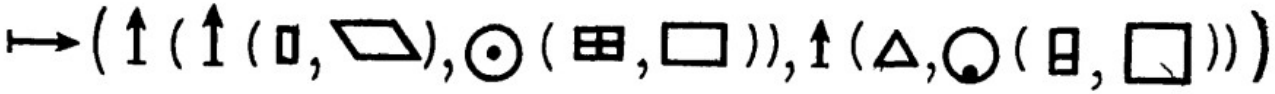
⟨профиль⟩ →  (⟨крыша⟩, ⟨стена⟩)

⟨домик⟩ → ⟨фасад⟩,

⟨домик⟩ →  (⟨домик⟩, ⟨профиль⟩).

Примеры двумерных грамматик...

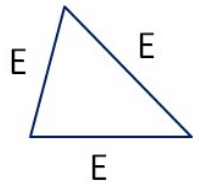
Описание трёх изображений:

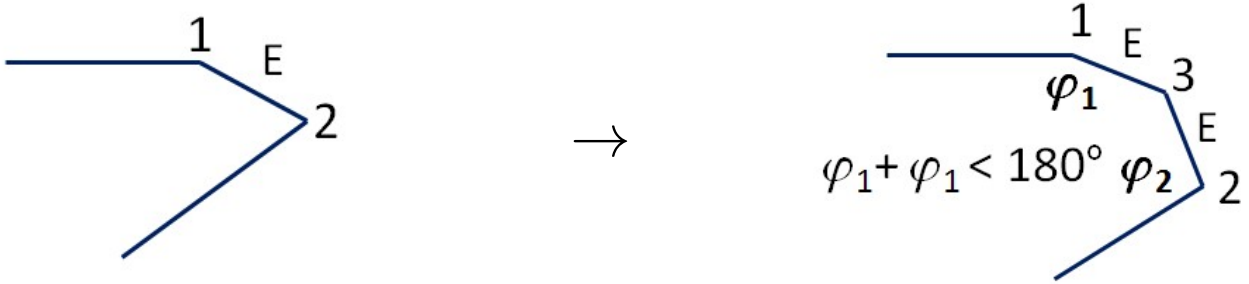
Домик	*Описание*
	
	
	

Примеры двумерных грамматик. 2. Многоугольники

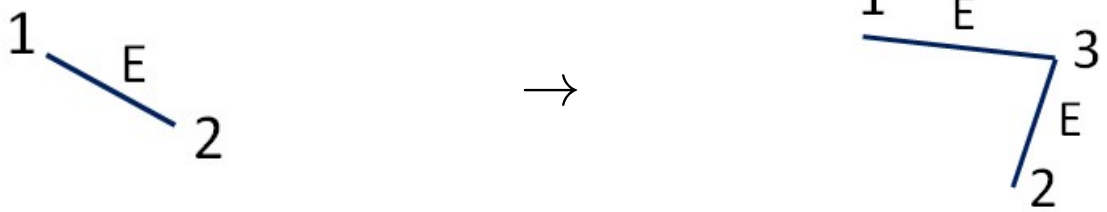
② Грамматика, описывающая многоугольники:

1) выпуклые —

$R: I \to$ 



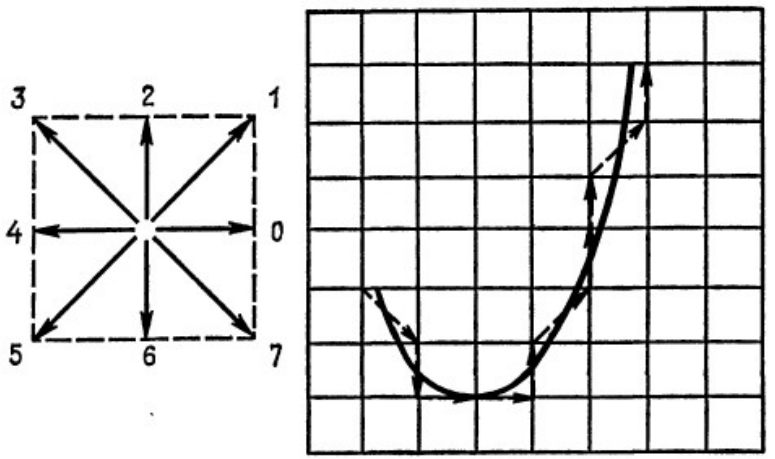
2) общего вида — замена последнего правила



Выделение терминальных элементов

Методы выделение терминальных элементов на изображении:
упор на 1) границы; 2) области.

1. Выделение терминальных элементов на основе границ — схема *цепного кодирования*:



- на изображение накладывают сетку и узлы сетки, наиболее близкие к точкам изображения соединяют отрезками прямых;
- каждому полученному отрезку присваивают восьмеричное число

— изображение представляется поледовательностью/ями восьмеричных чисел (7600212212).

Выделение терминальных элементов: цепное кодирование

Свойства цепного кодирования:

- поворот на угол, кратный 45°, сводится к $+_8$;
- легко выполнимы растяжение, определние длины кривой, самопресечений...
- применимо для границ произвольной связности.

Метод применялся для

- распознавания рукопечатного текста;
- классификации фотографий треков частиц в пузырьковой и искровой камерах;
- анализе хромосом;
- идентификации отпечатков пальцев
- ...

Выделение терминальных элементов

2. Выделение терминальных элементов на основе областей

Терминальные элементы — полуплоскости: выпуклые многоугольники представляются пересечением конечного числа полуплоскостей.

Полуплоскости $\rightarrow$ буквы,
Слова $\rightarrow$ выпуклые многоугольники,
Предложения $\rightarrow$ фигуры.

- $H_1, \ldots, H_k$ — полуплоскости с заданными заранее направлениями $\varphi_1, \ldots, \varphi_k$;
- G — группа параллельных переносов;
- первичное множество

$$R_j = \bigcap_{i=1}^{2k} g_i^j H_i,\ g_i^j \in G;$$

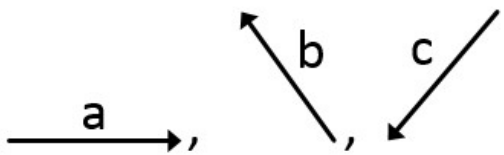
- многоугольник

$$A = \bigcup_{j=1}^{l} R_j.$$

Построение грамматики

Пример. Построить грамматики для регулярного языка $L = \{ a^n, b^n, c^n \mid 1 \leqslant n \leqslant 3 \}$, описывающий равносторонние треугольники с длиной стороны $1, 2, 3$.

1. Автоматная грамматика $G_3 = \langle V, W, I, R \rangle$, где
$W = \{ I, A_1, A_2, B_{10}, B_{20}, B_{30}, B_{21}, B_{31}, B_{32}, C_1, C_2, C_3 \}$,

$V = \{$ 

$\}$, R:

$$
\begin{array}{lll}
I \to aA_1 & B_{10} \to bC_1 & B_{32} \to bC_3 \\
I \to aB_{10} & B_{20} \to bB_{21} & C_1 \to c \\
A_1 \to aA_2 & B_{21} \to bC_2 & C_2 \to cC_1 \\
A_1 \to aB_{20} & B_{30} \to bB_{31} & C_3 \to cC_2 \\
A_2 \to aB_{30} & B_{31} \to bB_{32} &
\end{array}
$$

Построение грамматики...

2. Бесконтекстная грамматика $G_2 = \langle V, W, I, R \rangle$, где $W = \{ I, A_1, A_2, B_1, B_2, B_3, C \}$, $V = \{ a, b, c \}$, R:

$$
\begin{array}{lll}
I \to aA_1C & A_1 \to aA_2C & B_2 \to bB_1 \\
A_1 \to b & A_2 \to aB_3C & B_1 \to b \\
A_1 \to aB_2C & B_3 \to bB_2 & C \to c
\end{array}
$$

Беконтекстная грамматика G_2 гораздо компактнее автоматной G_3.

Грамматика непосредственно оставляющих для данного языка (G_1, не приведена) мало отличается от беконтекстной G_2.

Построение грамматики...

3) Программная грамматика $G_p = \langle V, W, I, P, J \rangle$, где $W = \{I, B, C\}$, $V = \{a, b, c\}$, $J = \{1, \ldots, 5\}$, P:

Метка	Ядро	I	F
1	$I \to aB$	$\{2, 3\}$	$\{\varnothing\}$
2	$B \to aBB$	$\{2, 3\}$	$\{\varnothing\}$
3	$B \to C$	$\{4\}$	$\{5\}$
4	$C \to bC$	$\{3\}$	$\{\varnothing\}$
5	$C \to c$	$\{5\}$	$\{\varnothing\}$

Все правила имеют вид $A \to \beta$, $A \in W$, $\beta \in T^*$, такая программная грамматика называется *бесконтекстной*.
Беконтекстная программная грамматика G_p описывает язык $L = \{a^n, b^n, c^n \mid n = 1, 2, \ldots\}$ и ещё более компактна, чем, беконтекстная G_2.

Разделы

Альтернативный способ задания автоматной грамматики

— конечным недетерминированным автоматом, генерирующим регулярный язык.

Определение

Конечный недетерминированный автомат есть пятёрка $\widetilde{\mathcal{A}} = \langle X, K, \varphi, P, \psi \rangle$, *где (все множества конечны):*

- X — *выходной алфавит (символы* X *генерируются автоматом* $\widetilde{\mathcal{A}}$*),*
- K — *множество состояний автомата,*
- φ — *предикат на* K*, определяющий начальные состояния,*
- ψ — *предикат на* K*, определяющий финальные состояния,*
- P — *предикат на* $K \times X \times K$*, определяющий многозначную функцию переходов: если в* $(i-1)$*-й момент времени автомат находился в состоянии* k' *и* $P(k', x, k) = 1$*, то в* i*-й момент автомат может выдать символ* x *и перейти в состояние* k*.*

Язык $L(\widetilde{\mathcal{A}})$ автомата $\widetilde{\mathcal{A}}$

— множество слов, которые могут появится на его выходе: $\overline{x} = (x_1, x_2, \dots x_n) \in L(\widetilde{\mathcal{A}})$, если существует такая последовательность $(k_0, k_1, \dots, k_n)$ состояний $\widetilde{\mathcal{A}}$, что

1) $\varphi(k_0) = 1$ (k_0 — начальное состояние);
2) $P(k_{i-1}, x_i, k_i) = 1$, $i = \overline{1,n}$ (переход $k_{i-1} \to k_i$ возможен для любого i);
3) $\psi(k_n) = 1$ (k_n — финальное состояние).

Это эквивалентно истинности предиката

$$F(\overline{x}) = \bigvee_{k_0 \in K} \dots \bigvee_{k_n \in K} \varphi(k_0) \,\&\, \mathop{\&}_{i=1}^{n} P(k_{i-1}, x_i, k_i) \,\&\, \psi(k_n),$$

где $F(\overline{x})$ — утверждение «*слово* $\overline{x} = (x_1, \dots, x_n)$ *принадлежит языку* $L(\widetilde{\mathcal{A}})$».

Компактная запись предиката $F(\overline{x})$

— в виде матричного произведения:

$\varphi = (\varphi_1, \ldots, \varphi_{|K|})$ — вектор-строка, $\varphi_k = \varphi(k)$,

$\psi = (\psi_1, \ldots, \psi_{|K|})$ — вектор-столбец, $\psi_k = \psi(k)$,

$P_i = \|P(k', x_i, k)\|$ — квадратная порядка $|K|$ матрица переходов, отвечающая выходному символу x_i, $i = \overline{1, n}$ (ясно, что существует всего $|X|$ различных матриц P_i),

$\otimes$ — матричное произведение с операциями сложения $\vee$ и умножения $\&$, тогда

$$F(\overline{x}) = \varphi \otimes \bigotimes_{i=1}^{n} P_i \otimes \psi.$$

Сложность вычисления этого предиката — $O(|K|^2 n)$.

Цель распознавания — определить $\overline{x} = (x_1, x_2, \ldots, x_n) \stackrel{?}{\in} L(\mathcal{A})$; для этого нужен *распознающий автомат*.

Задачу «$\overline{x} \in L(\widetilde{\mathcal{A}})$?» решает распознающий автомат

Определение

Конечный детерминированный автомат $\mathcal{A}$ *есть пятёрка* $\langle X, K, k_0, q, K' \rangle$, *где (все множества конечны)*

X — *входной алфавит (символы* X *подаются на вход автомата* $\mathcal{A}$*),*

K — *множество состояний автомата,*

$k_0 \in K$ — *одно из начальных состояний,*

$q : K \times X \to K$ — *функция переходов,*

$K' \subset K$ — *множество финальных состояний.*

Если в $(i-1)$-й момент времени $\mathcal{A}$ находился в состоянии k, а в i-й момент на его вход подан символ x, то $\mathcal{A}$ окажется в состоянии $q(x,k) \Rightarrow$ слово $x_1, x_2, \ldots, x_n$, поданное на вход $\mathcal{A}$ однозначно определит последовательность его состояний $k_0,\ k_1 = q(k_0, x_1),\ \ldots,\ k_n = q(k_{i-1}, x_n)$.

Автомат $\mathcal{A}$ распознаёт принадлежность слова $\overline{x}$ языку $L(\mathcal{A})$, если $k_n \in K'$.

Генерирующий автомат как грамматика

Рассмотрим генерирующий автомат $\widetilde{\mathcal{A}} = \langle X, K, \varphi, P, \psi \rangle$.

- Назовём X и K — *терминальным и нетерминальным алфавитами* соответственно.
- Функцию φ представим в виде подмножества начальных состояний (*аксиом*) $K^0 = \{ k \in K \mid \varphi(k) = 1 \}$.
- Функции P и ψ выразим в виде подмножеств троек (k', x, k'') и пар (k, x), $k, k', k'' \in K$, $x \in X$ соответственно.
- Если справедливо $P(k', x, k'') = 1$, то тройку (k', x, k'')
  - запишем в виде правила $k' \to xk''$,
  - а если при этом справедливо ещё и $\psi(k'') = 1$, то вводим ещё и правило $k' \to x$.

Множество правил обозначим R.

Генерирующий автомат как грамматика...

В результате автомат $\widetilde{\mathcal{A}}$ представляется в виде регулярной (автоматной) грамматики как четвёрка

$$G(\mathcal{A}) = \langle X, K, K^0, R \rangle.$$

Регулярная грамматика определяет язык, т.е.сть $L \subset X^*$, следующим образом:

1) слово, состоящая из единственного символа, который является одной из аксиом, называется выведенным в данной грамматике;
2) если слово $\overline{x}k'$, $\overline{x} \in X^*$, $k' \in K$ выведено, а множество правил содержит правило $k' \to x'k''$, то слово $\overline{x}x'k''$ тоже считается выведенным;
2) если слово $\overline{x}k'$, $\overline{x} \in X^*$, $k' \in K$ выведено, а множество правил содержит правило $k' \to x'$, то слово $\overline{x}x'$ принадлежит языку L, порождаемому данной грамматикой.

Генерирующий автомат как грамматика...

Автомат $\widetilde{\mathcal{A}}$ и грамматика $G(\widetilde{\mathcal{A}})$ порождают один и тот же язык $L \subset X^*$ (X^* — рефлексивное замыкание алфавита X = множество всех предложений конечной длины, включая пустое, в алфавите X).

Возможен и обратный переход — от регулярной грамматики к автомату.

Недетерминированный конечный автомат и регулярная грамматика — два эквивалентных способа задания регулярных языков (типа 3).

Операции на множестве языков

Введем следующие три операции на множестве языков.

1. *Итерация языка* L есть язык, обозначаемый L^* и определяемый следующим образом.
   - пустое слово (слово нулевой длины), принадлежит L^*.
   - если слово $\overline{x}$ принадлежит L^*, слово $\overline{y}$ принадлежит L, то слово $\overline{x}\,\overline{y}$ также принадлежит L^*.

2. *Конкатенация языков* L_1 *и* L_2 есть язык, обозначаемый $L_1 L_2$ и содержащий все слова вида $\overline{x}\,\overline{y}$, $\overline{x} \in L_1$, $\overline{y} \in L_2$ и только их.

3. *Объединение языков* L_1 *и* L_2 есть язык $L_1 \cup L_2$.

Регулярные выражения в алфавите X I

Регулярные языки можно также задать с помощью *регулярных выражений* по следующим правилам.

1. $\varnothing$ — регулярное выражение, обозначающее пустое множество слов.
2. $\#$ — регулярное выражения, обозначающее язык, состоящий из единственного пустого слова (нулевой длины).
3. Для каждого символа $x \in X$ запись x — регулярное выражение, обозначающее язык, состоящий из единственного однобуквенного слова x.
4. Если α — регулярное выражение языка L, то $(\alpha)^*$ — регулярное выражение для итерации языка L^*.

Регулярные выражения в алфавите X II

5. Если α_1 и α_2 — регулярные выражение языков L_1 и L_2 соответственно, то $\alpha_1\alpha_2$ — регулярное выражение для конкатенации L_1L_2, а α_1, α_2 — регулярное выражение для объединения $L_1 \cup L_2$.

Например, $a(b, c)^*$ — регулярное выражение, задающее множество слов

1) начинающихся символом a,
2) вслед за которым идёт любая (возможно пустая) последовательность, составленная из символов b и c.

Пример записи регулярного языка

Задача: задать язык L, множество слов вида

$$\langle \textit{идентификатор} \rangle = \langle \text{«}\textit{сумма произведений идентификатов}\text{»} \rangle,$$

в алфавите $X = \{a, b, c, +, \times, =\}$, где *идентификатор* — слово длины > 0, состоящее букв a, b, c — простейшие операторы присваивания.

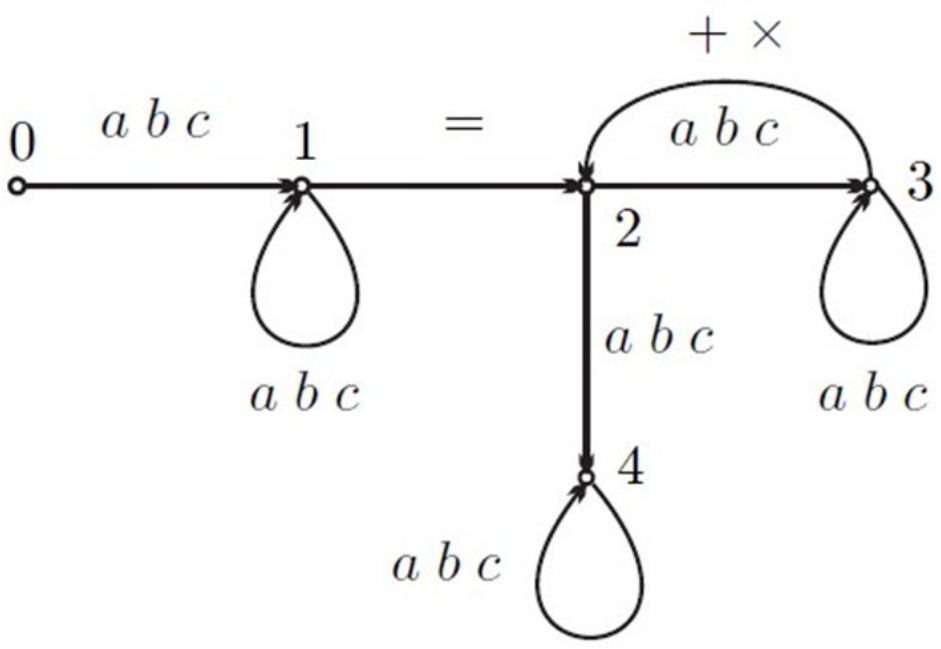
Например $x = ab + c \times a + aba \in L$, а $bc =, a = a + \times b \notin L$.

① Регулярное выражение, задающее язык L:

$$(a, b, c)(a, b, c)^* = ((a, b, c)(a, b, c)^*(+, \times))^* (a, b, c)(a, b, c)^*.$$

Пример записи регулярного языка...

② Недетерминированный автомат, порождающий язык L:



Выходной алфавит $X = \{a, b, c, +, \times, =\}$.
Множество состояний $K = \{0, 1, 2, 3, 4\}$ — множество вершин графа; начальное состояние — 0, конечное — 4.

Пример записи регулярного языка...

Если стрелке $k' \to k''$ приписан символ x, то $P(k', x, k'') = 1$ (для всех других троек функция $P = 0$), т.е. P принимает значение 1 на следующих тройках:

$(0, a, 1)$, $(0, b, 1)$, $(0, c, 1)$,
$(1, a, 1)$, $(1, b, 1)$, $(1, c, 1)$,
$(1, =, 2)$,
$(2, a, 3)$, $(2, b, 3)$, $(2, c, 3)$,
$(3, a, 3)$, $(3, b, 3)$, $(3, c, 3)$,
$(3, +, 2)$, $(3, \times, 2)$,
$(2, a, 4)$, $(2, b, 4)$, $(2, c, 4)$,
$(4, a, 4)$, $(4, b, 4)$, $(4, c, 4)$.

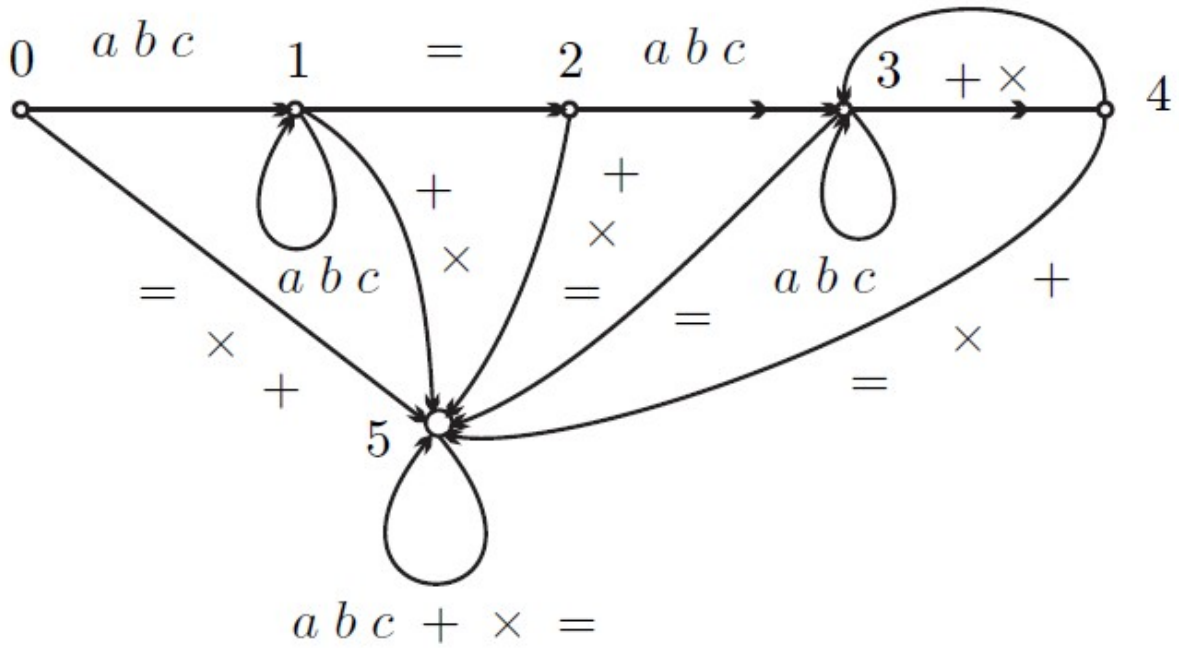
Пример записи регулярного языка...

③ Язык L как регулярная грамматика:
терминальный алфавит $V = \{a, b, c, =, +, \times\}$,
нетерминальный алфавит $W = \{0, 1, 2, 3, 4\}$,
начальное состояние $I = 0$, правила R:

$$0 \to a1,\ 0 \to b1,\ 0 \to c1, \qquad 2 \to a4,\ 2 \to b4,\ 2 \to c4,$$
$$1 \to = 2, \qquad 4 \to a4,\ 4 \to b4,\ 4 \to c4,$$
$$2 \to a3,\ 2 \to b3,\ 2 \to c3, \qquad 4 \to a,\ 4 \to b,\ 4 \to c,$$
$$3 \to a3,\ 3 \to b3,\ 3 \to c3.$$

Пример записи регулярного языка...

④ Детерминированный автомат, распознающий язык L:



Начальное состояние $k_0 = 0$, финальное $K' = \{3\}$ (5 — поглощающее состояние).

Стрелки задают функцию переходов: если стрелка начинается в вершине k', заканчивается в вершине k'' и на ней записан символ x, это обозначает, что $q(k', x) = k''$.

Пример записи регулярного языка...

Входной алфавит $X = \{a, b, c, =, +, \times\}$,
множество состояний $K = \{0, \ldots, 5\}$.

Любое правильное слово переводит автомат в состояние 3, а любое неправильное — в какое-то другое, что достигается выбором функции переходов q.

В рамках структурного подхода задача распознавания состоит в построении алгоритма, который для любого слова $\overline{x}$ данного регулярного языка определяет, принадлежит ли оно языку L.

Эта задача оказывается не очень трудной, если регулярный язык выражен с помощью генерирующего автомата: распознающий автомат определяется через множество последовательностей, для которых матричное произведение равно 1.

Штрафные автоматы и языки

Для формализации сложных понятий (классов) необходимо задание функции, выражающей степень уверенности принадлежности объекта классу $\Rightarrow$ понятие штрафных автоматов и языков.

Пусть X и K — два конечных множества, а три функции

$$\varphi : K \to \mathbb{R}_{\geqslant 0}, \quad P : K \times X \times K \to \mathbb{R}_{\geqslant 0}, \quad \psi : K \to \mathbb{R}_{\geqslant 0}$$

определяют поведение *штрафного автомата* $\mathcal{F}$:

- автомат может начать свою работу в любом состоянии $k \in K$, заплатив при этом штраф $\varphi(k)$;
- если автомат в $(i-1)$-й момент находился в состоянии k', то в i-й момент он может сгенерировать символ x и перейти в состояние k, заплатив штраф $P(k', x, k)$;
- автомат может завершить работу в любой момент i, заплатив за это штраф $\psi(k_i)$.

Штрафные автоматы и языки...

Штрафной язык определяется автоматом $\mathcal{F} = \langle X, K, \varphi, P, \psi \rangle$ и числом ε: в него входят те и только слова, штраф за генерирование которых не превосходит ε.

Регулярные языки — собственное подмножество штрафных.

Множество языков, которые могут быть определены указанным способом, включают все регулярные языки, а также некоторые другие языки, не являющиеся регулярными. Распознавание принадлежности последовательности $\overline{x} = (x_1, x_2, \ldots, x_n)$ заданному штрафному языку сводится к вычислению предиката

$$\min_{k_0} \min_{k_1} \ldots \min_{k_n} \left\{ \varphi(k_0) + \sum_{i=1}^{n} P(k_{i-1},\, x_i,\, k_i) + \psi(k_n) \right\} \overset{?}{\leqslant} \varepsilon.$$

Распознавание принадлежности последовательности штрафному языку

Полукольцо — алгебраическая структура, похожая на кольцо, но без требования существования противоположного по сложению элемента.

Пример полукольца — $(\min, +)$; здесь сложение — $\min$, умножение — $+$.
Обозначим в этом полукольце операцию матричного произведения $\odot$.

Тогда та же формула запишется как:

$$\varphi \odot \bigodot_{i=1}^{n} P_i \odot \psi \overset{?}{\leqslant} \varepsilon.$$

Задача остается простой: её формулировка указывает алгоритм решения с вычислительной сложностью $O(|K|^2 n)$.